Assignment 1 (15 Marks-7.5%)

SEC NO	Date for Showing the output to Instructor	Due Date for final submission through CMS
1	14th October 11:00 AM only	By 15/10/2020 11:00 PM

 # Delay on showing the output or submission will lose 20% for every one-day delay from the due date.

Team Details:

Terminal No: 3

Sl No	Name	ID No
1	Kaushik Kalakonda	2020A3PS0468H

Problem Statement: Design a state machine to control the tail lights of a 1965 Ford Thunder bird (Figure 1). The tail lights are composed of three lights on each side which operate for the turns as shown in figure 2. The state machine has two inputs **(LEFT, RIGHT)** and 6 outputs **(LC, LB, LA, RA, RB and RC)**. When **(RIGHT=0 and LEFT=0)** or when **(RIGHT=1 and LEFT=1)** no lights will turn ON. If **(RIGHT=0 and LEFT=1)** then lights LC, LB, and LA will be ON as shown in figure 2(a) indicating **LEFT** turn. If **(RIGHT=1 and LEFT=0)** then lights RA, RB, and RC will be ON as shown in figure 2(b) indicating **RIGHT** turn. In addition to LEFT and RIGHT there are two more inputs **Clk** and **Reset** for normal operation of FSM. When **Reset** is enabled all lights will be OFF**.** The flashing rate of LEDs is 2Hz (or 4Hz) (i.e. the time between two successive states is 0.5s (or 0.25s)).

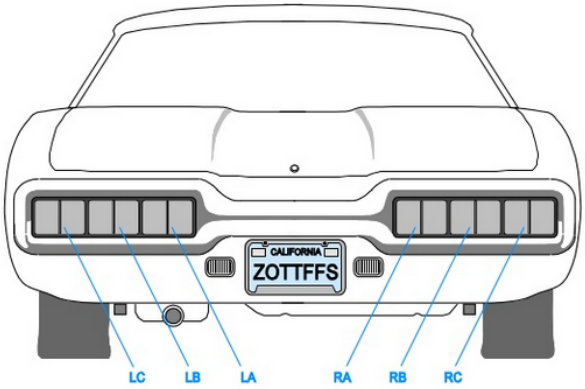


Figure 1

Left Turn **Right Turn**

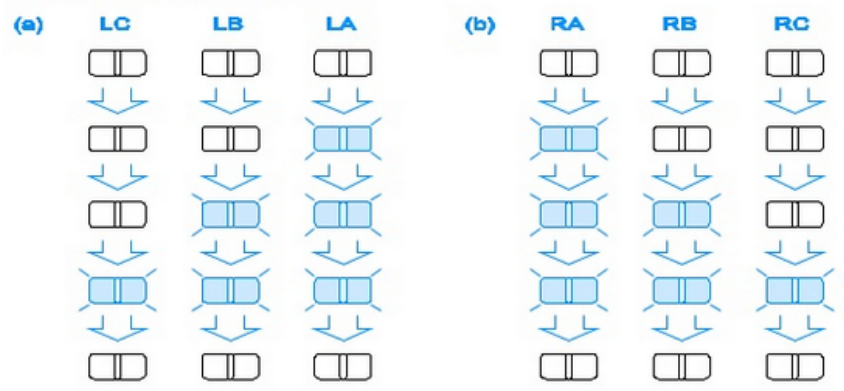


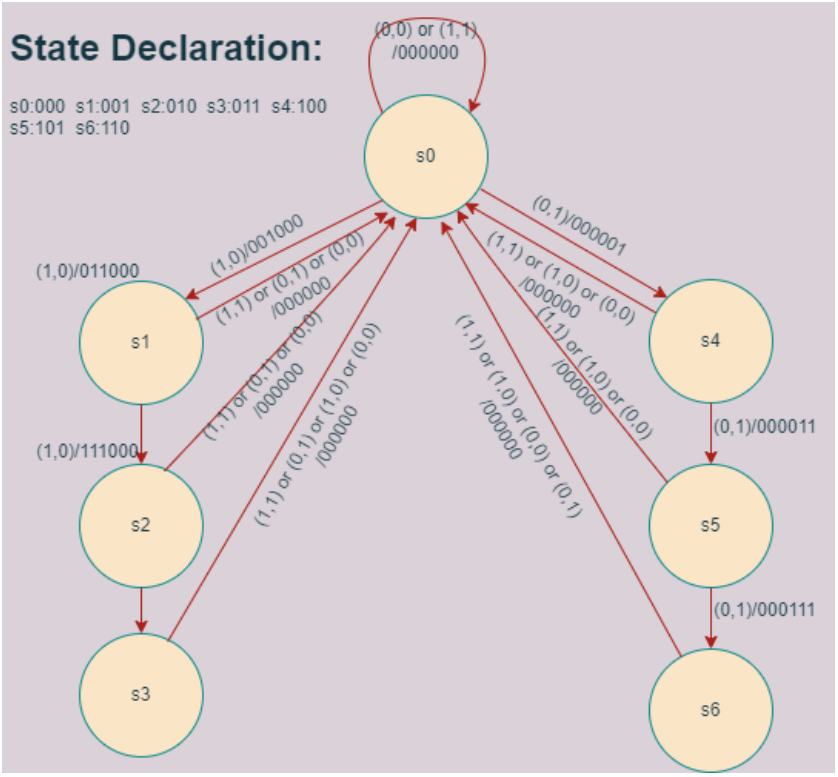
Figure 2(a) **Figure 2(b)**

Figure 2

(Please refer to the file named "**Vivado_Design_Flow_All_Steps.pdf**" for a review of all the steps in the design flow)

1. **<u>Question</u>: Draw the FSM (can be an image) with proper description.**

Answer:



2. Create a Vivado Project and write Verilog code (**Car_FSM.v** with comments) for implementing the above FSM.

   <u>Question</u>: Paste the image of verilog code Car_FSM.v.

Answer:

```
`timescale 1ns / 1ps

module Car_FSM(

input clk,
input reset,
input left,
input right,
output reg [2:0] l,
output reg [2:0] r
    );

//clock divisions for 1 MHz / 2^27 = 2-4 Hz
reg [26:0] clk_div;
always@(posedge clk, negedge reset)
begin
if(reset==1)
clk_div=0;
else
clk_div=clk_div+1;
end

//present and next states
reg [2:0] ps, ns;

//states of the fsm
parameter s0 = 3'b000,s1 = 3'b001,s2 = 3'b010,s3 = 3'b011,s4 = 3'b100,s5 = 3'b101,s6 = 3'b110;

always @(posedge clk_div[26], negedge reset)
    begin
    if (reset ==1)
    ps=s0;
    else
    ps <= ns;
    end

//following the flow model of the fsm
always @(ps, left, right)
    begin
    case (ps)
        s0 : begin
```

```
    case (ps)
        s0 : begin
            if (left==0 && right==1)
            ns = s1;
            else if (left==1 && right==0)
            ns = s4;
            else
            ns = s0;
            end
        s1 : begin
            if (left==0 && right==1)
            ns = s2;
            else
            ns = s0;
            end
        s2 : begin
            if (left==0 && right==1)
            ns = s3;
            else
            ns = s0;
            end
        s3 : begin
            ns = s0;
            end
        s4 : begin
            if (left==1 && right==0)
            ns = s5;
            else
            ns = s0;
            end
        s5 : begin
            if (left==1 && right==0)
            ns = s6;
            else
            ns = s0;
            end
        s6 : begin
            ns = s0;
            end
    endcase
    end
```

```
81
82  //output of the lights at the present state of the fsm
83  always @(ps)
84      case (ps)
85      s0 : begin
86          l = 3'b000;
87          r = 3'b000;
88      end
89      s1 : begin
90          l = 3'b000;
91          r = 3'b001;
92      end
93      s2 : begin
94          l = 3'b000;
95          r = 3'b011;
96          end
97      s3 : begin
98          l = 3'b000;
99          r = 3'b111;
100         end
101     s4 : begin
102         l = 3'b001;
103         r = 3'b000;
104     end
105     s5 : begin
106         l = 3'b011;
107         r = 3'b000;
108     end
109     s6 : begin
110         l = 3'b111;
111         r = 3'b000;
112     end
113     endcase
114
115 endmodule
116
```

3. Write the test bench **Test_Car_FSM.v** and simulate your design to check the functionality.

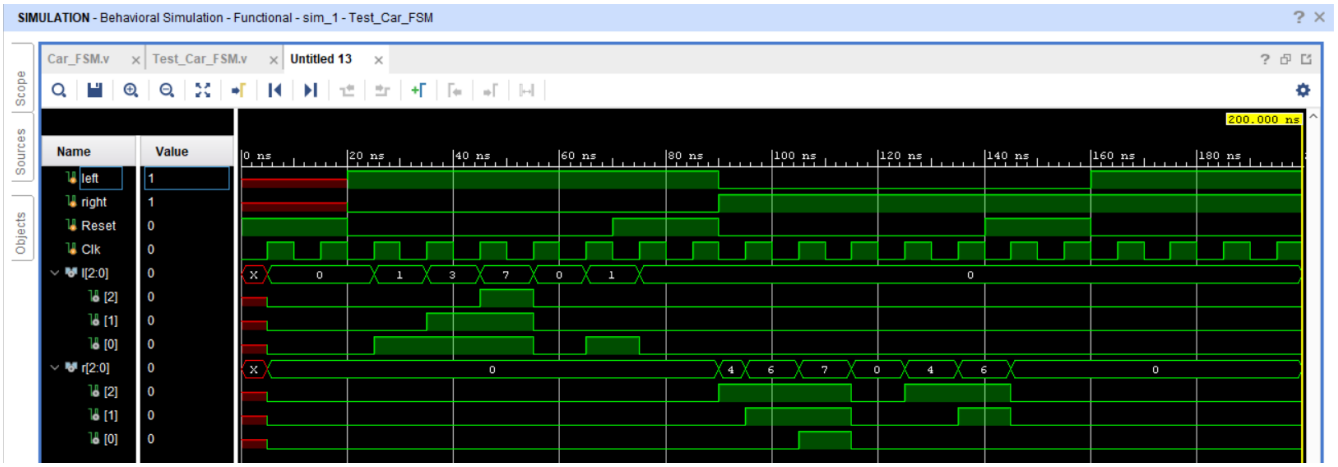
   <u>Question</u>: Paste the image of test bench verilog code Test_Car_FSM.v.

Answer:

```
module Test_Car_FSM(
    );
    reg left;
    reg right;
    reg Reset;
    reg Clk;
    wire [2:0] l;
    wire [2:0] r;

    Car_FSM a1(Clk, Reset,left, right, l,r);
    initial
    begin
    Clk=0;
    repeat(50)
    #5 Clk=~Clk;
    $finish;
    end

    initial
    begin
    Reset=1'b1;#20
    left=1'b1;right=1'b0;
    Reset=1'b0;
    #50 Reset=1'b1;
    #20 Reset=1'b0;
    left=1'b0;right=1'b1;
    #50 Reset=1'b1;
    #20 Reset=1'b0;
    left=1'b1;right=1'b1;
    #50 Reset=1'b1;
    #20 Reset=1'b0;
    left=1'b0;right=1'b0;
    end

endmodule
```

<u>Question</u>: Paste the image showing the simulated waveforms for FSM (Behavioral Simulation). Clearly show the LEFT turn and RIGHT turn Cases.

Answer:



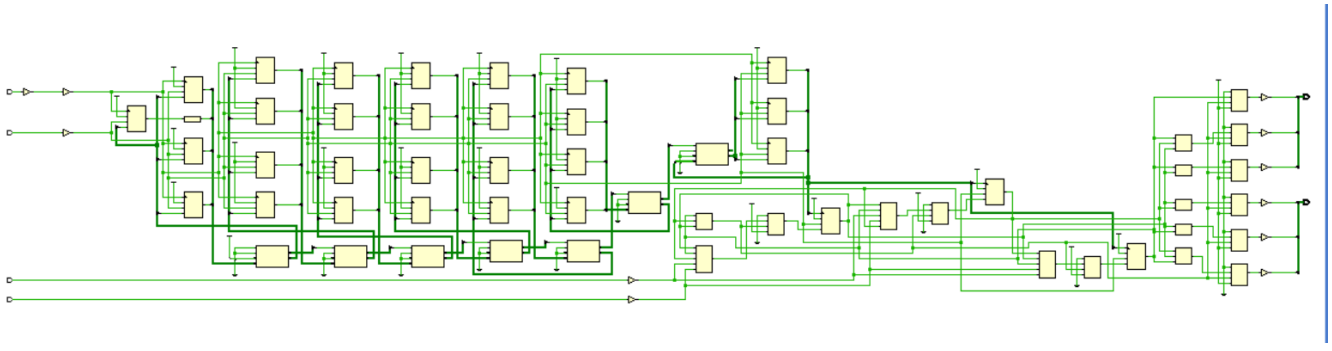
4. Add clock division code to **Car_FSM.v** such that the actual input **Clk (Y9 pin with frequency of 100MHz)** is converted to Clock of frequency 2Hz (or 4Hz). This 2Hz (or 4Hz) signal is used as clock for running the FSM. Connect the new (2Hz or 4Hz)) clock as output of **Car_FSM.v** for reference.

5. Plan your I/O mapping (using **I/O planning** option) such that actual input **Clk** is connected to internal clock pin **Y9**, **Reset** is connected to push button switch, other inputs (**LEFT** and **RIGHT**) are connected to DIP switches and outputs are connected to LEDs. In the ZedBoard, the pin numbers indicating the DIP switches, LEDs and internal clock are listed in table uploaded in CMS. Save the mapping information as **Car_FSM.xdc**.

6. Synthesize (**Run Synthesis**).

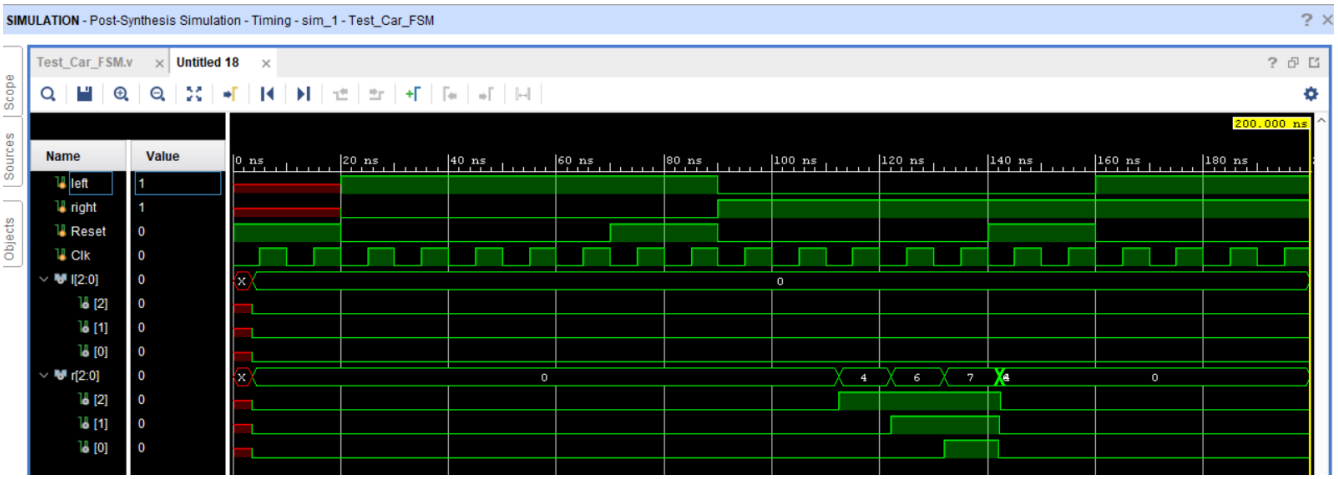
   <u>Question</u>: Paste the image showing the schematic after synthesis.

   Answer:



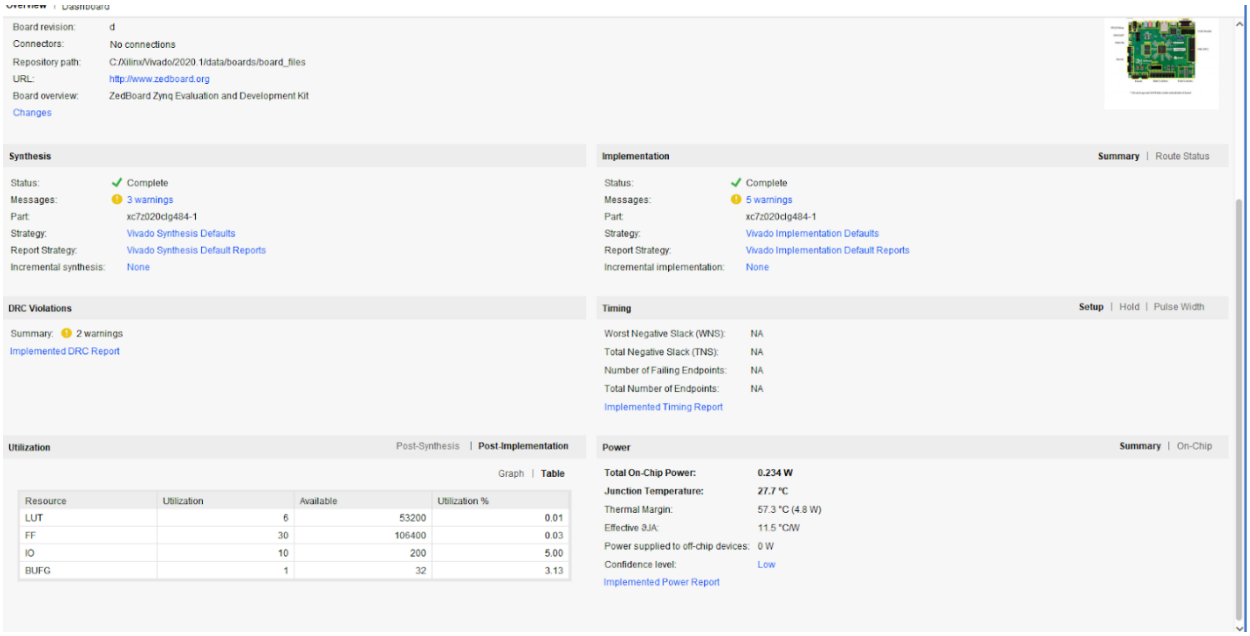
   <u>Question</u>: Paste the image showing the Post-Synthesis Simulation for FSM.

Answer:



<u>Question</u>: Check the summary report and report hardware utilization for the FSM implementation.

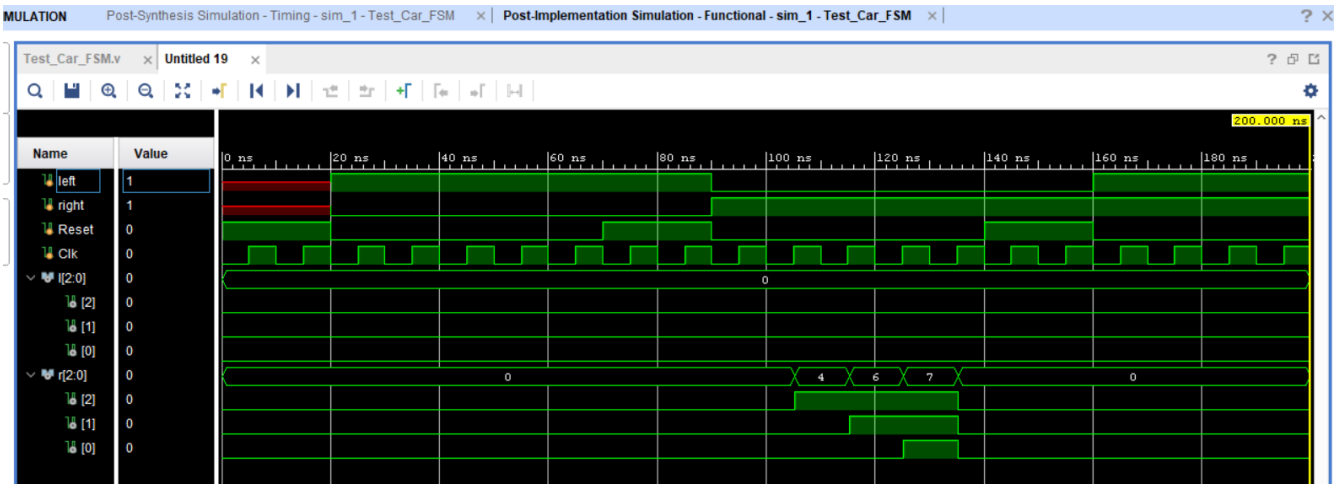
Answer:



7. Implement the design (**Run Implementation**).

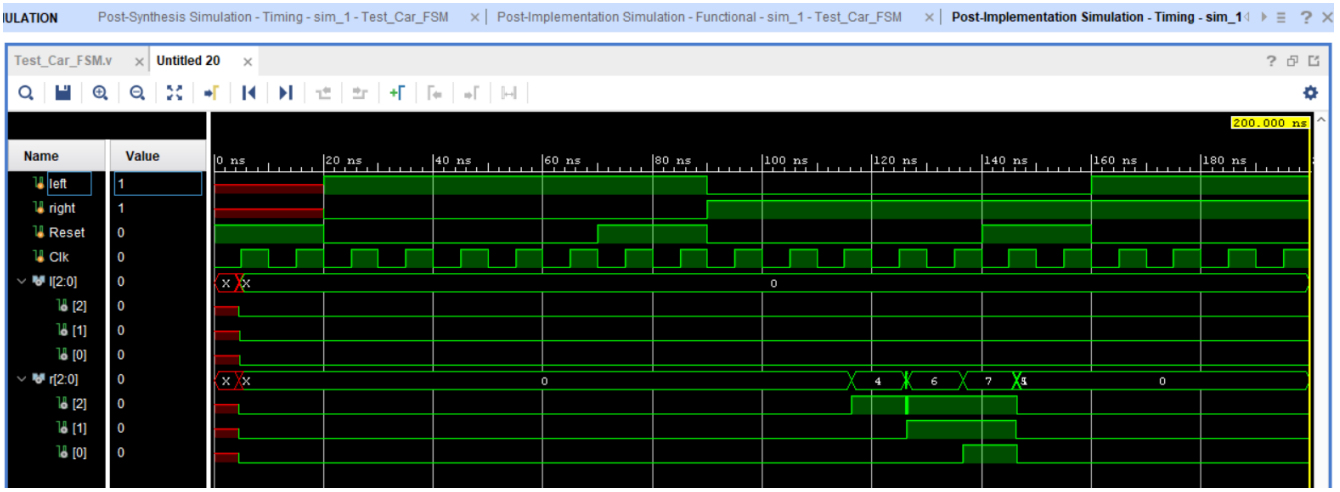
<u>Question</u>: Paste the image showing the Post-Implementation Simulation for FSM.

Answer:



<u>Question</u>: Paste the image showing the Post-Implementation timing Simulation for FSM.

Answer:



<u>Question</u>: Justify the difference between Post-Implementation and Post-Implementation timing Simulation waveforms.

Answer: Timing simulation is to verify whether the circuit will work at the required speed. Functional simulation is to verify the design of our circuit.

8. **Generate Bitstream** and port your design on to FPGA (**Open Hardware Manager**□ **New Target**□… **Program Device**)

9. **Check the output on FPGA.**

10. Show the output to the instructor.

11. Submit following files as a Zipped folder with file name as <Student1_ID_No>_<Student2_ID_No>.zip through CMS before due date.

1) Completed Document

2) Car_FSM.v (with proper comments)

3) Test_Car_FSM.v (with proper comments)

4) Car_FSM.xdc.